
ERDC MSRC/PET TR/01-04

# FORTRAN 77 to FORTRAN 90
# Source Code Conversion and Maintenance Tools

by

Richard Weed

28 February 2001

02h00301

**Work funded by the DoD High Performance Computing Modernization Program ERDC Major Shared Resource Center through**

Programming Environment and Training (PET)

Supported by Contract Number: DAHC94-96-C0002
CSC Nichols

# FORTRAN 77 to FORTRAN 90 Source Code Conversion and Maintenance Tools

Richard Weed *
Mississippi State University

## 1 Introduction

For most users, converting a large legacy FORTRAN 77 code to FORTRAN 90 is a daunting task. Most legacy FORTRAN code is written in UPPER CASE format that is difficult to read for new users more familiar with C or FORTRAN 90 code. In additon, the fixed field length of legacy FORTRAN 77 code makes code modification more difficult than the free field source format of FORTRAN 90. To assist the Army Engineer Research and Development Center(ERDC) Major Shared Resource Center(MSRC) users in converting from FORTRAN 77 to FORTRAN 90, a small package of FORTRAN and C routines has been developed. The routines described in this report are designed to ease some of the burden of converting FORTRAN 77 codes to FORTRAN 90 and to assist in the overall code maintenance and documentation task.

The FORTRAN and C routines in this package are designed to assist in the conversion of FORTRAN 77 fixed format source to FORTRAN 90 free (or fixed) formats. In addition, some small C utilities that are callable from FORTRAN are included. The C routines can be used in other applications. The following sections give descriptions of the three primary source conversion and maintenance codes in the package along with usage information and instructions on how to build the codes on a variety of platforms. Examples of the output from the primary conversion utility along with information on how to use the C utilities in other codes are given in the Appendices.

## 2 Fortran Source Conversion and Maintenance Codes

### 2.1 *ftof90.f90*

*ftof90* is the main program in the package. Its principal function is to convert FORTRAN 77 fixed length source, written in all CAPS, to a more readable (for most people) lower case

*ERDC MSRC PET CSM Onsite Lead, Engineer Research and Development Center, ATTN: CEERD-IH-C, 3909 Halls Ferry Road, Vicksburg, MS 39180. E-mail: rweed@erdc.hpc.mil

format that can be used as FORTRAN 90 free field or fixed field source [1]. It is NOT a complete F77 to F90 conversion routine. It does not realign *do* and *if* blocks and replace *continue* statements with *enddo*s etc. See the *convert.f90* program by Metcalf and Reid [2] for a conversion routine that can realign code blocks. Another feature of *ftof90* is its ability to process ALL of the files in a directory with a given file extension.

*ftof90* will perform the following code conversions:

1. Convert all UPPER CASE text to lower case, except text between accepted string delimiters (' or "). The code will detect strings broken across continuation lines. The default is to convert to lower case, but this can be overriden by user input. (Option)

2. Reformat default and/or user defined keywords to either a Leading Cap format (i.e. Read, Do, etc.) or an ALL CAP format. Note: this option turns on lower case conversion if it was not selected. The default is no conversion. (Option)

3. Reformat continuations to F90 free field form with all continuation characters in column six reset to an &. An ampersand is added to either column 73 or the current end of line plus two characters (except when an embedded comment card is detected) of the card that preceeds the current continuation. This conforms to the F90 style of continuations. Placing the & in columns 73 through 80 will allow the code to be used as either fixed or free format. The code checks for tab characters in columns one through six during its check for existing continuation characters. However, no attempt is made to expand and remove the tabs. The default is no conversion. (Option)

   WARNING: The current code will NOT handle the cases a comment card between continuations or continuations inside *cpp* #ifdef blocks. If a target code has one or more of these conditions then modification of continuations to F90 format should not be selected. However, changing the current column six character to an & can still be performed. If F90 free field format is needed, continuation modification can still be selected. Just be prepared to edit individual files to correct the problems that arise from embedded comments. A warning message will be printed for each file that contains these conditions.

4. Modify F77 style relational operators (i.e. .eq., .ne., .ge., .le., .lt., .gt.) to the F90 C-like forms (i.e. ==, /=, >=, <=, <, >). This feature also turns on lower case conversion. The default is no conversion. (Option)

5. Change comment characters in column one from F77 style to a !. The default is to change the comments, but this can be overridden by user input. (Option)

6. Remove MS-DOS ^M (EOL) and ^Z (EOF) characters. (Always done)

7. Strip trailing blanks to reduce file size. (Always done)

### 2.1.1 Using *ftof90*

*ftof90* is an interactive program that prompts for user input. Most of the input requires either a *yes* (*y*) or *no* (*n* or *enter*) response. The responses for an entire session can be saved to a script file so that the user can perform the same conversions on other sets of files without having to reenter the responses by hand. The response file is read from standard input so the user need only type *ftof90 <script.txt*, where *script.txt* is the name of the saved script file, to execute the program without reentering the responses interactively.

*ftof90* will process either a user specified single file or ALL of the files with a specified file extension in the the current directory. *ftof90* assumes that the default input file extension is *.f* and the default output file extension for the converted file is *.f90*. This can be changed by user input. If the specified input and output file extensions are the same, a "_new" is appended to the output filename (i.e. the converted version of *file.f* will be named *file_new.f*). This option allows the user to perform code conversions on *.f90* files that are written in all caps or generate a fixed field file with a *.f* extension for use with the *xlf* compiler on IBM SPs and SMPs.

The default keywords are defined in *default_keywords.f90*. The file *new_keywords.txt* included in the package illustrates how additional keywords can be defined by the user. The code has fixed left and right conditions that must be met before a string is accepted as a keyword. The values in *default_keywords.f90* represent the most common control and type definition keywords. Intrinsic functions such as *tan*, *sin*, etc. are not defined but can be included in the user's *new_keywords.txt* file. The user has the option of replacing the default keywords with the contents of *new_keywords.txt* or appending the contents to the existing default keyword list. A listing of the current *default_keywords.f90* routine that sets the default keywords is given in Appendix C.

An example of a typical execution of *ftof90* is given in Appendix A. The sample FORTRAN 77 code given in Appendix A.1 was converted into the FORTRAN 90 free field format code shown in Appendix A.2. The interactive session that generated the FORTRAN 90 code is shown in Appendix A.3. These data illustrate the ability of *ftof90* to generate more readable FORTRAN 90 source from legacy FORTRAN 77 code without changing the meaning of the original source.

### 2.1.2 Sample *new_keywords.txt* file

The following illustrates the format of the *new_keyword.txt* file. The first input is the number of entries in the file. Each subsequent line is a free field format character variable.

*new_keywords.txt*:

```
23
'elseif'
'Elseif'
'select'
'case'
'default'
'contains'
'module'
'public'
'private'
'Enddo'
'Endif'
'call'
'atan'
'asin'
'acos'
'tan'
'sin'
'cos'
'mod'
'ichar'
'achar'
'char'
'exit'
```

### 2.1.3 Subroutines Called by *ftof90*

*ftof90* calls the following subroutines:

1. *mod_keywords.f90*: Modifies keywords to either leading cap or all cap format.
2. *mod_relationals.f90*: Modifies relational operators.
3. *default_keywords.f90*: Sets default keyword values.
4. *blank_card.f90*: Sets a character string of a given length to all blanks.
5. *com_or_cpp.f90*: Determines if the current line is a comment or a *cpp* directive.

## 2.2 *dos2unx.f90*

*dos2unx.f90* is a standalone program that will strip the MS-DOS EOL and EOF marks (^M and ^Z) from a source file or ALL of the files with a given extension in the current directory. Most current UNIX machines will have a system MS-DOS to Unix conversion

routine available. However, these system routines usually work on only one file at a time. Another advantage of this code is that it is transportable to any machine with F90 and C compilers. The output files will have filenames consisting of the old filename and extension with a "_unx" appended to the file name (i.e. *file.f* becomes *file_unx.f*). The primary advantage of *dos2unx* over system routines, such as the *to_unix* routine found on SGI systems, is its ability to process all of the source files with a given file extension in the current directory.

### 2.3 *printps.f90*

*printps.f90* will generate Postscript listings in two column landscape format (i.e. 2up format) of all of the files with a specified extension in the current directory. It REQUIRES a Unix *csh* shell script file named *lp2upf* that calls a system text to Postscript converter such as *lptops* on SGI machines or the GNU *enscript* [3] routine. Sample scripts for *lptops* and *enscript* are included in the software package. As with *dos2unx*, the primary advantage of *printps* is its ability to process all of the file with a given file extension in the current directory. Individual files can be processed by executing the *lp2upf* script on the selected file, i.e. *lp2upf file.f* will generate a Postscript file named *file.f.ps*. See Appendix C for an example of *printps/lp2upf* output using the *enscript* routine. The following lines illustrate the contents of the *lp2upf* script using *lptops* and *enscript*.

*lptops*:

```
#! /bin/csh -f
/usr/lib/print/lptops -FCB -H -G -M2 -P7pt -O1pt -U \$1 | cat >$1.ps
```

*enscript*:

```
#! /bin/csh -f
/usr/bin/enscript -G2r -L60 -f Courier-Bold7 -F Courier-Bold12 -p $1.ps $1
```

## 3 C Utility Routines

The following C utility programs are included in this package. They are designed to be called from FORTRAN 77 or FORTRAN 90 and can be used on all of the ERDC MSRC systems. The utilities are as follows:

1. *syscall.c*: A FORTRAN callable wrapper around the C *system()* function. Executes a system shell command passed as a character string from FORTRAN.

2. *homedir.c*: Returns the user's home directory as set by the HOME environment variable.

3. *currentdir.c*: Returns the current directory.

4. *getenvf.c*: Returns the value set for an environment variable. The variable name is passed as a character string.

These routines can be compiled and linked seperately with the user's FORTRAN 90 code. An example FORTRAN 90 routine that illustrates how the C utilities can be called from FORTRAN is given in Appendix B.

## 4 Building the Source Utilities

The user must have both a F90/F95 compiler and a C compiler available to build *ftof90*, *dos2unx*, and *printps* along with *make*. The routines are built using *make* on most machines by simply typing *make* while inside the current directory. The make file structure is set up to define system specific information (i.e. compiler names, flags, etc.) by using a shell script run from inside *make* (*make_defaults*) to build a *make* include file that defines the appropriate data. *make_defaults* uses the *uname* function to define the system architecture. Values for the following systems are set by default: SGI IRIX and IRIX64 systems, IBM AIX systems, CRAY T3E systems, and Red Hat Linux systems using *gcc* and the Lahey F95 Express compiler. The makefile sets generic Unix values if one of the default machines is not detected. *make_defaults* should be modified as needed for other machines. Copy one of the existing *Elseif* sections and modify as needed for a particular architecture.

NOTE: The target operating systems for this software are Unix and Unix-like operating systems such as Linux and FreeBSD. The current makefile will not work on MS-DOS/WINDOWS. The code should be compiled by hand and modified as needed for non-Unix machines. The *syscall* strings in *ftof90*, *dos2unx*, and *printps* will have to be modified to be consistent with MS-DOS commands (i.e. rm to del, ls to dir, etc.) See *Makefile_utils* for the subroutine dependencies.

## 5 General Code Usage

As stated earlier, the codes are interactive and prompt the user for the correct input. Most responses require a *y* for *yes* or some other character (*n* or just an enter/return) for *no*. When prompted for a file extension, type the extension WITHOUT the leading period (i.e. enter *f* not *.f*, etc.). All of the code conversion utilities create new output files. Therefore, the original source files are never changed or overwritten.

## 6 Code Validation and Verification

The utilities described in the package have executed successfully on a wide range of architectures. The routines were verified with both test data such as the *testsrc.f* file found in the software package and real legacy codes. Although every attempt was made to anticipate

potential problems that might arise in legacy codes, it is possible that some legacy codes will be encountered that have code structures that *ftof90* and the other codes in this package will not process correctly. Therefore, any problems encountered with using the codes in this package should be addressed to the author. In addition, please read the standard disclaimers contained in the source code before using any of the routines in this package.

## 7 Obtaining the Code, Who to Contact, Etc.

ERDC MSRC users can access the source code and compiled binaries for the utilities via the */usr/local/usp/PETtools* directory that exists on each of the ERDC MSRC HPC systems. The source code, make files, PDF and Postscript copies of this report, and the executable files are contained in the *Srcutils* subdirectory. Requests for the code package from outside the ERDC MSRC user community should be sent to the author through the ERDC MSRC Customer Assistance Center via email addressed to info-hpc@erdc.hpc.mil. Questions about the code and bug reports can be sent directly to the author via email addressed to rweed@erdc.hpc.mil.

## Acknowledgments

This work was funded inpart by the DoD High Performance Computing Modernization Program ERDC MSRC through Programming Environment and Training (PET), Contract Number: DAHC 94-96-C0002, Computer Sciences Corporation. Views, opinions, or findings contained in this report are those of the authors and should not be construed as an official Department of Defense position, policy, or decision unless so designated by other official documentation. Permission to publish this paper is granted by the Headquarters, U.S. Army Corps of Engineers.

## References

[1] Ellis, T.M.R., Phillips, I.R., Lahey, T.M. *FORTRAN 90 Programming*. Addison-Wesley, Inc., Reading, MA, 1994.

[2] Metcalf, M. and Reid, J.K., *FORTRAN 90 Explained*. Oxford University Press, Oxford, England, 1990.

[3] Anon. http://www.gnu.org

# APPENDIX A. Test Program Input and Example Output

## A.1 *ftof90* Test Program Input - *testsrc.f*

```
      PROGRAM TESTSRC
C
!    This is a test OF THE ftof90 UtIlItY
!    The following is NOT a real program so
!    DON't try to compile and run
C XXXXXXXXX - Strings and comments keep same case
C
      REAL AA(100)
      CHARActer*3 ans
#ifdef (TEST)
      AB=b
#else
      AB=c
#endif
#ifdef (TEST2)
C
*
C
  !   test ''' TEST
C       Test of continuation in comments
C    &  test
C
        TAB=Char(9)
      IF (ans.eq.'Y' .OR. ANS.eq."y") THEN
       ANS = 'MYANS'
      ELSEIF
      string2 = "TEST of'Tt' StRinG
     & "
      End If
      If (ans.eq.'y')CALL A
       IF (b.eq.c) Then
         b=ab
       End IF
      If (b.eq.ab) goto 3
      call BBB
      WRite(*,'('' test of continuations   ! trailing comment
     ; continuation 1
     2 Continuation 2
     3 '')')
      Write(*,'(" A1, F10, L3",'' A=b '',
     & i3, "c=d",
     & A6,)') A, B, C
       Y=ab+b /
     ;  C
       b = tan(ab) + sin(b) + tan(d)+tan(abc)
c
      End
```

### A.2 Output from *ftof90* - *testsrc.f90*

```
      Program testsrc
!
!    This is a test OF THE ftof90 UtIlItY
!    The following is NOT a real program so
!    DON't try to compile and run
! XXXXXXXXX - Strings and comments keep same case
!
      Real aa(100)
      Character*3 ans
#ifdef (TEST)
      ab=b
#else
      ab=c
#endif
#ifdef (TEST2)
!
!
!
  !   test ''' TEST
!       Test of continuation in comments
!    &  test
!
        tab=Char(9)
      If (ans == 'Y' .or. ans == "y") Then
       ans = 'MYANS'
      ElseIf
      string2 = "TEST of'Tt' StRinG &
     & "
      End If
      If (ans == 'y')Call a
       If (b == c) Then
         b=ab
       End If
      If (b == ab) GoTo 3
      Call bbb
      Write(*,'('' test of continuations  &! trailing comment
     & continuation 1 &
     & Continuation 2 &
     & '')')
      Write(*,'(" A1, F10, L3",'' A=b '', &
     & i3, "c=d", &
     & A6,)') a, b, c
       y=ab+b / &
     &  c
       b = Tan(ab) + Sin(b) + Tan(d)+Tan(abc)
!
      End
```

### A.3 Typical Interactive Session for *ftof90*

The following is a reconstruction of the complete interactive session used to generate the output shown in Appendix A.2.

```
            ****** Welcome to ftof90 ******

      Answer the following with a Y or y for yes and
      either an n or carraige return (enter) for no


  Default file extensions are f (input) and f90 (output)- Change-(y/n) : n
  Modify all files in current dir-(y/n) : n
  Enter file name of file to modify : testsrc.f
  Source code is converted to lower case by default
  Do you wish to keep old file case format-(y/n) : n
  Convert comments to lower case (Default is no) (y/n) : n
  Modify FORTRAN Keywords to Leading or ALL Cap Format-(y/n) : y
  Default keyword format is Leading Caps. Change to all CAPS (y/n) : n
  Modify F77 relationals to F90 forms (ie .eq. to == etc.)-(y/n) : y
  F77 style comments converted to ! by default
  Keep F77 style comments (c,C or *) - (y/n) : n
  Default line length is 80 charactes- modify-(y/n) : n
  Modify continuations to F90 Free field format-(y/n) : y
  Current column six characters will be changed to an &
  This will handle the case of continuing a character string by default

  Default location of free field continuation & is current end of line
  plus two characters. Placing the & in column 73 will allow you to use the
  same source for both fixed and free formats

  Change default location of ampersand to column 73 (y/n) : n
  Checking for a new_keywords.txt file

  Keyword file found - Replace(r) or append(a) to defaults (r/a) : a
  Save response script to file convert_script.txt (y/n) : y
    Converting testsrc.f to testsrc.f90
```

# APPENDIX B. C Utilities Fortran 90 Example

```
      Program testcutils
!
!     Tests C utilities
!
      Implicit NONE ! Force explicit type definition
!
      Integer, Parameter :: max_len=51
      Character(LEN=17) :: cmdstring
      Character(LEN=31) :: home_name
      Character(LEN=51) :: current_name
      Character(LEN=max_len) :: workdir, envval
      Integer :: ret,ie
!
      Integer, External :: getenvf ! The following need to be externals
      Integer, External :: homedir
      Integer, External :: currentdir
!
!  sycall executes the shell command string cmdstring
!
      cmdstring = "ls -1 *.f >ls.out"
      Call syscall(cmdstring) ! lists of all .f files into file ls.out
!
!  homedir is called from FORTRAN as follows and returns the users
!  home directory as set by the environment variable $HOME
!
      ret=homedir(home_name) ! ret = length of homename
      Print *,' Home Directory Name =', home_name(1:ret)
!
!  currentdir is called from FORTRAN as follows and returns the users
!  current directory
!
      ret=currentdir(current_name) ! ret = length of current_name
      Print *,' Current Directory Name =', current_name(1:ret)
!
!  getenvf is called from FORTRAN as follows and returns the value
!  set for the environment variable $WORKDIR
!
      envval = 'WORKDIR'
      ie = LEN_TRIM(envval)
      ret=getenvf(envval(1:ie),workdir) ! ret = length of WORKDIR string
      Print *,' $WORKDIR =',workdir(1:ret)
!
      Stop
!
      End Program testcutils
```

# APPENDIX C. *printps/lp2upf* Listing of *default_keys.f90*

The listing on the following page illustrates the output generated by the GNU *enscript* routine using the parameters defined previously for the *lp2upf* script file. Similar output can be generated by *lptops*. This file also shows the current default values for FORTRAN 77 and FORTRAN 90 keywords used by *ftof90*.

```
      Subroutine Default_keywords(keyword,kword_length,nkeys, &
                                  kwmax)
!
!  Programmed by Richard Weed, Miss. State University
!
!   Set default keywords and string length. Assumes keywords
!   in input file were set to all lower case by ftof90.
!   The program checks for the left and right conditions that
!   signify a key_word. Both conditions must be met before
!   the string is accepted as a keyword. Therefore, you do
!   NOT need to use spaces etc. to delimit strings. Subroutine
!   mod_keywords will check for the correct left and right
!   conditions
!
!  Used by ftof90.f90
!
!
!**                          Authors Disclaimer                          **
!**                                                                      **
!**  This codes was developed under U.S. Government funding and is released  **
!**  to the public under the terms of the following disclaimer. The code     **
!**  works as advertised but no guarantee is made that it is 100 per cente   **
!**  free of bugs. Therefore, you accept the full risk and responsibility    **
!**  for any damages that occur from using the code. Use of any part of      **
!**  this code is taken by the author to be an implicit agreement of these   **
!**  terms.                                                                  **
!**                       U.S. Government Disclaimer                         **
!**                                                                      **
!**  This program is furnished by the U.S. Army Engineer Research and        **
!**  Development Center, Major Shared Resource Center (ERDC MSRC) "as is "   **
!**  and is accepted and used by the recipient with the express              **
!**  understanding that the Government makes no warranties, expressed or     **
!**  implied, concerning the accuracy, completeness, reliability, usability  **
!**  or suitability for any particular purpose of the information and data   **
!**  within this program or furnished in connection therewith, and the       **
!**  Government shall be under no liability whatsoever to any person by      **
!**  reason of any use made thereof. This program belongs to the U.S.        **
!**  Government; therefore, the recipient further agrees not to assert any   **
!**  proprietary rights therein or to represent the source code as belonging **
!**  to anyone other the U.S. Government.                                    **
!**
!
! Code Begins HERE
!
      Implicit NONE
!
      Integer  :: nkeys, kwmax, j
      Character(LEN=*) :: keyword(kwmax)
      Integer :: kword_length(kwmax)
!
!  The following are the most common control and type
!  definition keywords. Define specific intrinsic functions
!  (tan, cos, char, atan, etc) as needed in new_keywords.txt
!
      nkeys = 45
      keyword(1) = 'subroutine'
      keyword(2) = 'if'
      keyword(3) = 'do'
      keyword(4) = 'continue'
      keyword(5) = 'endif'
      keyword(6) = 'enddo'
      keyword(7) = 'end'
      keyword(8) = 'dimension'
      keyword(9) =  'equiva'
      keyword(10) =  'function'
      keyword(11) =  'call'
      keyword(12) =  'return'
      keyword(13) =  'parameter'
      keyword(14) =  'inclu'
      keyword(15) =  'real'
      keyword(16) =  'integer'
      keyword(17) =  'implicit'
      keyword(18) =  'write'
      keyword(19) =  'read'
      keyword(20) =  'print'
      keyword(21) =  'common'
      keyword(22) =  'then'
      keyword(23) =  'else'
      keyword(24) =  'stop'
      keyword(25) =  'goto'
      keyword(26) =  'data'
      keyword(27) =  'double'
      keyword(28) =  'precision'
      keyword(29) =  'float'
      keyword(30) =  'namelist'
      keyword(31) =  'character'
      keyword(32) =  'logical'
      keyword(33) =  'format'
      keyword(34) =  'open'
      keyword(35) =  'close'
      keyword(36) =  'while'
      keyword(37) =  'external'
      keyword(38) =  'intrinsic'
      keyword(39) = ' complex'
      keyword(40) = ' entry'
      keyword(41) = ' program'
      keyword(42) =  'pointer'
      keyword(43) =  'go to'
      keyword(44) =  'go'
      keyword(45) =  'to'
!
!     Get Keyword Lengths to last non blank character
!
      Do j=1,nkeys
        kword_length(j) = LEN_TRIM(keyword(j))
      EndDo
!
      End Subroutine Default_keywords
```